energies

MDPI

Article

Secure Elliptic Curve Crypto-Processor for Real-Time IoT Applications

Stefano Di Matteo *,†, Luca Baldanzi †, Luca Crocetti †, Pietro Nannipieri †, Luca Fanucci † and Sergio Saponara †

Department of Information Engineering, University of Pisa, Via G. Caruso 16, 56122 Pisa, Italy; luca.baldanzi@ing.unipi.it (L.B.); luca.crocetti@phd.unipi.it (L.C.); pietro.nannipieri@ing.unipi.it (P.N.); luca.fanucci@unipi.it (L.F.); sergio.saponara@unipi.it (S.S.)

* Correspondence: stefano.dimatteo@phd.unipi.it; Tel.: +39-050-27660

† These authors contributed equally to this work.

Abstract: Cybersecurity is a critical issue for Real-Time IoT applications since high performance and low latencies are required, along with security requirements to protect the large number of attack surfaces to which IoT devices are exposed. Elliptic Curve Cryptography (ECC) is largely adopted in an IoT context to provide security services such as key-exchange and digital signature. For Real-Time IoT applications, hardware acceleration for ECC-based algorithms can be mandatory to meet low-latency and low-power/energy requirements. In this paper, we propose a fast and configurable hardware accelerator for NIST P-256/-521 elliptic curves, developed in the context of the European Processor Initiative. The proposed architecture supports the most used cryptography schemes based on ECC such as Elliptic Curve Digital Signature Algorithm (ECDSA), Elliptic Curve Integrated Encryption Scheme (ECIES), Elliptic Curve Diffie-Hellman (ECDH) and Elliptic Curve Menezes-Qu-Vanstone (ECMQV). A modified version of Double-And-Add-Always algorithm for Point Multiplication has been proposed, which allows the execution of Point Addition and Doubling operations concurrently and implements countermeasures against power and timing attacks. A simulated approach to extract power traces has been used to assess the effectiveness of the proposed algorithm compared to classical algorithms for Point Multiplication. A constant-time version of the Shamir's Trick has been adopted to speed-up the Double-Point Multiplication and modular inversion is executed using Fermat's Little Theorem, reusing the internal modular multipliers. The accelerator has been verified on a Xilinx ZCU106 development board and synthesized on both 45 nm and 7 nm Standard-Cell technologies.

Keywords: Elliptic Curve Cryptography; European Processor Initiative (EPI); cryptography; Real-Time; ASIC; 7 nm; FPGA; verification; side-channel

Citation: Di Matteo, S.; Baldanzi, L.; Crocetti, L.; Nannipieri, P.; Fanucci, L.; Saponara, S. Secure Elliptic Curve Crypto-Processor for Real-Time IoT Applications. *Energies* **2021**, *14*, 4676. https://doi.org/10.3390/en14154676

Academic Editor: Manuel Moreno

Received: 6 July 2021
Accepted: 27 July 2021
Published: 1 August 2021

1. Introduction

Nowadays the request for secure communication over a network is growing dramatically. Different areas such as automotive, Internet of Things (IoT), health-care, storage and financial services require the exchange of sensitive information on insecure channels. Symmetric and asymmetric cryptography can provide several security services as authentication, key exchange, digital signature and data encryption, ensuring the protection of data exchanged. Elliptic Curve Cryptography (ECC) is a kind of asymmetric cryptography, which provides the advantage of obtaining an equivalent security level key size that is smaller in respect to other public key algorithms, such as Rivest-Shamir-Adleman (RSA) [1] or schemes based on the Discrete Logarithm Problem (DLP) [2,3]. ECC was introduced by Victor Miller [4] and Neil Koblitz [5] in 1985 and has been adopted by many standardization institutes such as IEEE [6], NIST [7], ANSI [8] and SECG [9].

The main operation involved in every cryptography scheme based on ECC is the Point Multiplication (PM), also named Scalar Multiplication (SM). Given two points Q and G

belonging to an elliptic curve and a scalar k, PM is denoted as $Q = kG$ and represents the sum of G to itself $(k-1)$ times to obtain the point Q. In ECC the point Q assumes the meaning of public key while k is the private key. The mathematical security of ECC relies on the hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP), that is, the problem of finding the value of k given the values of Q and G. In addition, the shape of the elliptic curve and its parameters must be properly selected in order to ensure the security and robustness of the whole system based on ECC. In 1999 NIST standardized five elliptic curves over a prime finite field $GF(p)$ [10], named NIST P-224, P-256, P-384 and P-521, that are widely used in many internet protocols and applications such as SSL (Secure Sockets Layer), TLS (Transport Layer Security) [11] and IPSec (IP Security) [12] and in some standards for automotive communication such as WAVE [13] and ETSI [14].

ECC algorithms can be implemented in software providing higher level of configurability respect to hardware solutions; however, hardware implementations can be suitable for particular scenarios such as power/energy and resource constrained devices or real time IoT applications. Work in [15] proposes a token-based security protocol for IoT devices that makes use of on-chip physically unclonable functions and ECC to authenticate devices in large-sized networks. The paper focuses on trading-off energy/quality of the protocol, and it shows that the required energy for executing the protocol is largely dominated by the ECC computation. The scheme proposed in [16] for Edge Computing and IoT is based also on ECC. In contexts like these, dedicated hardware for ECC could improve the performance in terms of speed/energy of the entire protocol. As stated in [17,18], for some markets (e.g., high-performance computing, automotive and Real-Time applications in general) hardware accelerators solutions could be mandatory in cases of ECC-based algorithms (e.g., ECDSA) due to their long execution times on low-power processors [18], or high energy consumption on general-purpose processors [17]. These remarks lead to the conclusion that hardware acceleration for ECC-based cryptographic algorithms seems to be mandatory for Real-Time IoT applications which simultaneously require high performance, low latency and limited power and energy consumption. This problem has been addressed by different researchers. The work in [19] is a review paper that shows some guidelines to aid hardware designers in choosing the combination of methods and algorithms for different application classes. Works like [20–24] focus on the acceleration of ECC.

In this paper, we propose a hardware architecture, configurable at synthesis level, to support NIST P-256 only, NIST P-521 only, or both the elliptic curves, which provides a security level from 128 to 256 bits. Such design is exploitable for accelerating the most used cryptographic schemes based on ECC. It makes use of a constant-time and Simple Power Analysis (SPA) resistant modified version of Double-and-Add algorithm to compute PM and of a constant-time version of Shamir's trick (algorithm 3.48 in [25]) to speed-up the Double Point Multiplication (DPM) required in ECDSA verification, using projective coordinates in order to avoid modular division. In addition, Fermat's Little Theorem has been adopted to reuse the internal modular multipliers avoiding to employ dedicated hardware for converting the coordinates in the standard domain. This work is part of the early development phase for the architecture of the ECC hardware accelerator within the Hardware Secure Module (HSM) of the European Processor Initiative [26] chip. NIST is running a standardization process for new public-key algorithmss, and currently ECC is adopted by several standards and the EPI project cryptographic functions based on ECC are required to provide the Root of Trust (RoT) of an EPI chip. The main contributions of this works are as follows:

- Architectural design of a configurable (at synthesis level) ECC crypto-processor for NIST P-256 and/or NIST P-521 elliptic curves, developed in the framework of the European Processor Initiative together with other cryptographic hardware accelerators (AES, RNG [27,28], SHA [29]). The proposed architecture supports the most used cryptographic schemes based on ECC such as ECDSA, ECDH, ECIES and ECMQV. The design is resistant to timing and SPA attacks and uses a constant-version of

Shamir's trick for Double Point Multiplication, and Fermat's Little Theorem to execute modular inversion.
- A simulated environment to extract and evaluate the power consumption of the circuit, which allowed the evaluation of the proposed countermeasure against SPA. The proposed simulated approach does not require any dedicated equipment to acquire power samples and can be adopted in the early design phase to help designers find effective architectural and algorithmic solutions against Power attacks.
- Synthesis on the open source 45 nm NANDGATE45 [30] library and 7 nm TSMC silicon technology (the first contribution available to the best of authors' knowledge) with a complete analysis of the performance in terms of complexity, throughput and power consumption.
- Verification and characterization in terms of resources utilization and throughput on a Xilinx ZCU106 development board equipped with Zynq UltraScale+ xczu7ev-ffvc1156-2-e MPSoC.

The remainder of this paper is organized as it follows: Section 3 discusses the preliminaries of ECC, PM and coordinates representation. Section 2 lists and analyzes the related works on hardware implementations of ECC accelerators. Section 4 presents the proposed hardware architecture and shows the FPGA verification approach. In addition, this section describes the power simulation environment we used to assess countermeasures against SPA and the power traces we extracted during PM operation; Section 5 shows the results of our design and a comparison with the state of the art. Finally, Section 6 concludes this paper.

2. Related Works

Several ECC hardware systems can be found in literature targeting high-performance, low-resources consumption or low-power. Research on hardware accelerators for ECC mainly focuses on improving the performance of PM operation, but sometimes no verification on side-channel attacks resistance is provided. The work in [31] is a Dual-Field ECC processor that exploits a hardware–software design to support an arbitrary elliptic curve. It adopts the radix-4 interleaved multiplication algorithm for modular multiplication and the Euclidean algorithm for modular inversion. To assess the resistance against power attacks, the authors implemented their design on a FPGA and recorded power traces by measuring the power consumption of the device. This work has been implemented in both XMC 55 nm CMOS technology and on Xilinx Virtex-4 FPGA platform. On CMOS technology it requires 1450 µs for a PM that is a relatively low speed. The design in [24] is the fastest we found in the literature; it adopts a full-word Montgomery multiplier and implements PA and PD operations concurrently. The authors synthesized their work in a 65 nm CMOS technology that requires only 12.5 µs for a PM, but the area consumption is extremely high. No consideration about side-channel resistance has been done. Their implementation is not suitable for resource-constrained devices and for IoT applications. The authors in [20] present a processor for NIST P-224 and P-256 elliptic curves. They used the Montgomery algorithm for modular multiplication, the binary inversion algorithm for modular inversion and Jacobian coordinates to represent the elliptic curve points. PM takes between 560 and 730 µs for 224-bit and 256-bit elliptic curves on a 65 nm CMOS technology. The area consumption is quite high and no side-channel countermeasures have been evaluated. The work in [32] is an elliptic curve processor over $GF(p)$ synthesized in TSMC 90 nm technology. The authors adopted a 3-pipelined-stage Montgomery multiplier and Standard projective coordinates, performing a PM in 120 µs with 540K gate counts. The authors propose a Montgomery ladder algorithm with a swap operation for PM and claim their solution is resistant to SPA attacks, but no experimental results have been provided to confirm this assumption. In [33] a cryptographic processor for general curves over $GF(p)$ is presented. The authors in this work employed a systolic arithmetic unit to implement the operations of addition, subtraction, multiplication and inversion, sharing the hardware resources and obtaining good performance in terms of area occupation but low speed. No

considerations about side-channel resistance are proposed. They synthesized the design on a 65 nm CMOS technology. The work in [21] is a ECC Processor for Weierstrass Curves over $GF(p)$ implemented on 7-series FPGA. It adopts a Montgomery multiplication which is constructed employing a large number of Digital Signal Processor (DSP) primitives. The PM is executed in constant time but no experimental verification about side-channel attacks resistance is provided. The performance in terms of speed is good but the resource consumption is very high. The work in [22] is a low hardware consumption design for elliptic curves from 160 to 256 bits over $GF(p)$. Interleaved Modular Multiplication and Binary Modular Inversion algorithms have been used, and the PM algorithm is claimed to be resistant to SPA attacks, but no experimental results have been provided. The design in [23] presents a novel modular squaring scheme that has been synthesized on a 130 nm CMOS technology. It reaches good performance in terms of area and speed, but no side-channel attacks resistance is guaranteed.

3. Preliminaries on ECC

3.1. Elliptic Curve Cryptography

An elliptic curve over prime field $GF(p)$ is defined by the Weierstrass equation:

$$y^2 = x^3 + ax + b \ (mod\ p) \tag{1}$$

where the parameters a and b are integers included in the prime field $GF(p)$ which satisfies $4a^3 + 27b^2 \neq 0 \ (mod\ p)$. A Weierstrass elliptic curve E over $GF(p)$ consists of a set of points $P = (x, y)$, with $x, y \in GF(p)$ together with an extra point O called "point at infinity". The NIST P elliptic curves are Weierstrass curves over $GF(p)$ and their parameters can be found in [9]. The set of elliptic curve points plus the point at infinity forms a group where the following group law can be defined:

- Point Addition (PA): $P_1(x_1, y_1) + P_2(x_2, y_2) = P_3(x_3, y_3)$ where

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \tag{2}$$

$$y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1 \tag{3}$$

- Point Doubling (PD): $2P_1(x_1, y_1) = P_3(x_3, y_3)$ where

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \tag{4}$$

$$y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1 \tag{5}$$

where $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ are two points on the elliptic curve. It should be noted that all the arithmetic operations (additions, subtractions, multiplications and divisions) described above are on the prime field $GF(p)$. PA and PD operations over such group are used to construct many elliptic curve crypto-systems, and a typical hierarchical structure of an ECC crypto-system is reported in Figure 1. At the top level there are protocols such as ECDSA, ECIES, ECDH and ECMQV. In the lower layer there are PM and DPM that will be discussed in Section 3.2; the next layer comprises the basic operations on the ECC points: PA and PD. They require the underlying level that consists of finite field arithmetic operations on $GF(p)$ such as modular addition, subtraction and multiplication. In our hierarchical structure we placed modular inversion at the same level of PA and PD because we implemented it using Fermat's Little theorem that exploits the operations at the lowest layer.

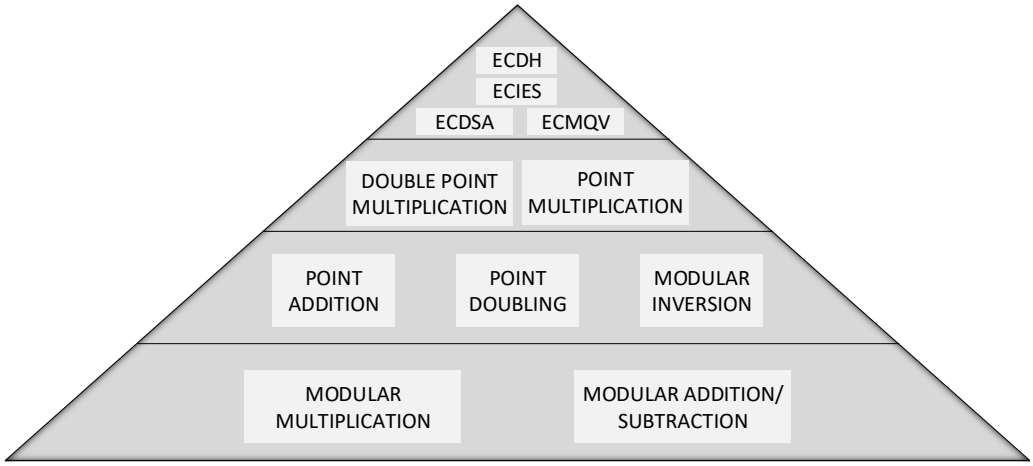

Figure 1. Implementation Hierarchy of the ECC Operations.

3.2. Point Multiplication and SPA

PM between an integer k and an elliptic curve point P is the main operation involved in every cryptographic scheme based on ECC. PM is indicated with $Q = kP$, and represents the sum of the point P to itself $(k-1)$ times:

$$Q = kP = \overbrace{P + P + \cdots + P}^{\text{(k - 1)}} \tag{6}$$

Many algorithms can be used to perform PM and the most know are based on Double-and-Add method. Algorithm 1 shows the Double-and-Add method for PM in Right-to-Left version.

Algorithm 1 Double-and-Add Right-to-Left.

Input: $P \in E, k = (k_{n-1}, k_{n-2}, .., k_1, k_0)$
Output: $Q = kP$

```
1: Q = O
2: R = P
3: for i = 0 to i = (n − 1) do
4:     if k_i = 1 then
5:         Q = Q + R
6:     end if
7:     R = 2R
8: end for
9: return Q
```

In Algorithm 1 the number of PA depends on the Hamming weight of k and its execution time is not fixed, making this kind of algorithm weak against timing attacks. In addition, the dissimilarity between PA and PD can be exploited by SPA [34] attacks. SPA involves the interpretation of power traces over time during the execution of PM in order to determine the integer k which in ECC must be secret. An attacker could easily extract the value of k by observing the power trace and understanding which operation the crypto-processor is performing. Timing attacks can be executed by an attacker at distance, by measuring the time needed to respond to a request, in contrast to SPA attacks that require physical access to the device equipped with the crypto-processor. In our work, we focused on protecting our ECC-system against both timing and SPA attacks. As reported in [35], the Double-and-Add-Always method can be used to hide the dependency of the key on the operations flow. As can be seen in Algorithm 2, in Double-and-Add-Always algorithm PA is executed even if the scalar bit is null and the result of this operation is discarded. This countermeasure theoretically does not allow to distinguish between real and dummy PA

operations. Nevertheless, in Algorithm 2 the presence of operations between real points and the point-at-infinity allows to guess part of the key k. At line 1 of the Algorithm 2 it can be seen that both the variables Q and T are initialized to the point-at-infinity, and in lines 4 and 5 they are summed to the variable R. As long as Q and T are equal to the point-at-infinity (this happens in the case of Q as long as no 1 is encountered in the key, and in the case of T as long as no 0 is encountered), the first PA operation between either Q and T and a real point can be identified, and an attacker may be able to assume some part of the key k. These considerations will be explained in more detail in Section 4.6, where the power traces extracted during the execution of Algorithm 2 will also be shown. To overcome this issue, our design implements a modified version of Algorithm 2, reported in Algorithm 3. The variable one_flag is used to indicate whether a 1 has been encountered in the key k. As long as one_flag is 0, dummy PA is executed between the input point P and R. When one_flag becomes 1, PA is performed between Q and R, and the resulting point is sampled only when a 1 is encountered. This method allows avoiding PA at point-at-infinity. We selected Right-to-left version Double-and-Add algorithms since they allow performing PA and PD operations simultaneously. Furthermore, our PM implementation does not use precomputations on the base point, and allows using different points P as input. ECDH, ECMQV and ECIES algorithms require the performance of PM between private and public keys where the public key could be different from the base point. In addition, we support the DPM operation required in ECDSA verification. DPM is composed of two separated PMs and one PA, as reported in the equation below:

$$Q = kP + lR \tag{7}$$

where Q,P and R are three different elliptic curve points while k and l are two scalars. A constant-time version of Shamir's trick is used in our implementation, reported in Algorithm 4.

Algorithm 2 Double-and-Add-Always Right-to-Left.

Input: $P \in E, k = (k_{n-1}, k_{n-2}, .., k_1, k_0)$
Output: $Q = kP$
1: $Q, T = O$
2: $R = P$
3: **for** $i = 0$ to $i = (n-1)$ **do**
4: **if** $k_i = 1$ **then**
5: $Q = Q + R$
6: **else**
7: $T = T + R$
8: **end if**
9: $R = 2R$
10: **end for**
11: **return** Q

Algorithm 3 Modified Double-and-Add-Always Right-to-Left.

Input: $P \in E, k = (k_{n-1}, k_{n-2}, .., k_1, k_0)$
Output: $Q = kP$

one_flag $= 0$, $R = 2P$
if $k_0 = 1$ **then**
    $Q = P$, one_flag $= 1$
end if
for $i = 1$ to $i = (n-1)$ **do**
    if $k_i = 1$ **then**
        if one_flag $= 1$ **then**
            $Q = Q + R$
        else
            $P + R$, $Q = R$
        end if
    else
        if one_flag $= 1$ **then**
            $Q + R$
        else
            $P + R$
        end if
    end if
    $R = 2R$
end for
return Q

Algorithm 4 Constant Time Version of Shamir's Trick.

Input: $P, R \in E, k = (k_{n-1}, .., k_0), l = (l_{n-1}, .., l_0)$
Output: $Q = kP + lR$

$S = P + Q$
$Q, T = O$
for $i = n - 1$ to $i = 0$ **do**
    $Q = 2Q$
    if $k_i = 1$ and $l_i = 1$ **then**
        $Q = Q + S$
    else if $k_i = 1$ and $l_i = 0$ **then**
        $Q = Q + P$
    else if $k_i = 0$ and $l_i = 1$ **then**
        $Q = Q + R$
    else
        $T = T + S$
    end if
end for
return Q

3.3. Coordinates Representation

PA and PD formulas reported in Section 3.1 can be used when the elliptic curve points are represented in the classical form (affine form). In this case both PA and PD require modular inversion on the prime field GF(p). However, since modular inversion is the most expensive finite field operation, a redundant projective representation can be used in order to avoid the modular inversion. Actually, only one modular inversion is needed to reconvert in affine coordinates. In projective representation, every point $P_1(x_1, y_1)$ can be mapped to $P_1(X_1, Y_1, Z_1)$ where Z_1 may be chosen arbitrarily. Selecting a projective representation, the form of Weierstrass equation, the points over the elliptic curve and the addition and doubling formulas change. The most common coordinate systems are

reported in Table 1, together with the number of modular multiplications and modular inversions required by each of them to compute PA and PD operations.

Table 1. Computational Cost of PA and PD.

	Affine	Standard Projectives	Jacobian Projectives
PA	3M + 1I	14M	16M
PD	4M + 1I	10M	8M

In this paper Standard projective coordinates are used. PA and PD in standard projective coordinates are reported respectively in equations 1 and 2. The addition between two points $P_1(X_1, Y_1, Z_1)$ and $P_2(X_2, Y_2, Z_2)$ is the point $P_3(X_3, Y_3, Z_3)$ such that:

$$\begin{cases} X_3 = BC \\ Y_3 = A(B^2X_1Z_2 - C) - B^3Y_1Z_2 \\ Z_3 = B^3Z_1Z_2 \end{cases} \tag{8}$$

where:

$$\begin{cases} A = Y_2Z_1 - Y_1Z_2 \\ B = X_2Z_1 - X_1Z_2 \\ C = A^2Z_1Z_2 - B^3 - 2B^2X_1Z_2 \end{cases} \tag{9}$$

The double of a point $P_1(X_1, Y_1, Z_1)$ is the point $P_3(X_3, Y_3, Z_3)$ such that:

$$\begin{cases} X_3 = EB \\ Y_3 = A(D - E) - 2C^2 \\ Z_3 = B^3 \end{cases} \tag{10}$$

where:

$$\begin{cases} A = 3(X_1 + Z_1)(X_1 - Z_1) \\ B = 2Y_1Z_1 \\ C = BY_1 \\ D = 2CX_1 \\ E = A^2 - 2D \end{cases} \tag{11}$$

4. Proposed Hardware Architecture

4.1. Modular Addition and Subtraction

Modular addition/subtraction algorithm is reported in Algorithm 5 where the steps 2–7 represent modular addition and the steps 10–15 represent modular subtraction. The hardware architecture is shown in Figure 2. In the case of modular addition $S = a + b(modp)$, the signal SEL_OP shall be set to 0; the first adder executes addition between the two inputs a, b providing sum S_1 and carry $Cout_1$, and the second one performs subtraction between S_1 and p with outputs S_2 and $Cout_2$. At the end, S_1 and S_2 are multiplexed according to line 4 of Algorithm 5. In the case of modular subtraction $S = a - b(modp)$, the signal SEL_OP shall be set to 1, and the first adder performs subtraction between the inputs a, b and the second one adds the result S_1 to the modulo p. Similarly to the first case, S_1 and S_2 are multiplexed according to to line 12 of Algorithm 5. This implementation requires one clock cycle.

Algorithm 5 Modular Addition/Subtraction.

Input: $a, b \in [0, p-1]$, p and SEL_OP
Output: $S = a \pm b (mod p)$

if $(SEL\_OP == 0)$ **then**
  $S_1 = a + b$
  $S_2 = S_1 + (\sim p) + (\sim SEL\_OP)$
  if $(Cout_1 | Cout_2)$ **then**
    $S = S_2$
  else
    $S = S_1$
  end if
else
  $S_1 = a + (\sim b) + SEL\_OP$
  $S_2 = S_1 + p$
  if $(Cout_1)$ **then**
    $S = S_2$
  else
    $S = S_1$
  end if
end if
return S

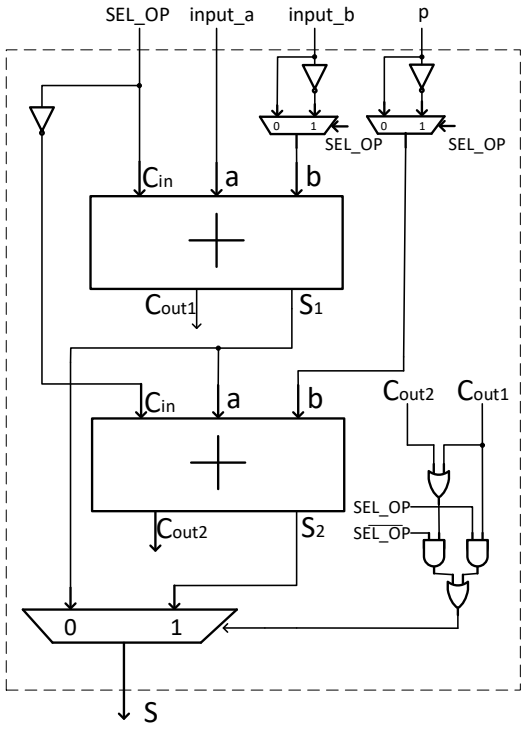

Figure 2. Modular Addition and Subtraction Architecture.

4.2. Modular Multiplication

As reported in Section 3.3 a projective representation of the elliptic curve can be used to avoid modular inversion in PA and PD formulas, at the cost of increasing the number of modular multiplications. For this reason, in the design of hardware architectures for ECC that adopt projective coordinates, the modular multiplier is the most important block. In this work we focused on NIST P-256/521 curves that support a fast modular reduction algorithm, reported in Algorithms 6 and 7.

Algorithm 6 Fast Modular Reduction for NIST P-256.

Input: $a = a_{15}2^{480} + a_{14}2^{448} + a_{13}2^{416} + a_{12}2^{384} + a_{11}2^{352} + a_{10}2^{320} + a_9 2^{288} + a_8 2^{256} + a_7 2^{224} + a_6 2^{192} + a_5 2^{160} + a_4 2^{128} + a_3 2^{96} + a_2 2^{64} + a_1 2^{32} + a_0$
Output: $r = a(mod p)$

1: $t = (a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0)$
2: $s1 = (a_{15}, a_{14}, a_{13}, a_{12}, a_{11}, 0, 0, 0)$
3: $s2 = (0, a_{15}, a_{14}, a_{13}, a_{12}, 0, 0, 0)$
4: $s3 = (a_{15}, a_{14}, 0, 0, 0, a_{10}, a_9, a_8)$
5: $s4 = (a_8, a_{13}, a_{15}, a_{14}, a_{13}, a_{11}, a_{10}, a_9)$
6: $d1 = (a_{10}, a_8, 0, 0, 0, a_{13}, a_{12}, a_{11})$
7: $d2 = (a_{11}, a_9, 0, 0, a_{15}, a_{14}, a_{13}, a_{12})$
8: $d3 = (a_{12}, 0, a_{10}, a_9, a_8, a_{15}, a_{14}, a_{13})$
9: $d4 = (a_{13}, 0, a_{11}, a_{10}, a_9, 0, a_{15}, a_{14})$
10: **return** $r = (t + 2s1 + 2s2 + s3 + s4 - d1 - d2 - d3 - d4) mod p$

Algorithm 7 Fast Modular Reduction for NIST P-521.

Input: $a = a_1 2^{521} + a_0$
Output: $r = a(mod p)$

1: **return** $r = (a_1 + a_0) mod p$

Thanks to these reduction algorithms, the multiply-then-reduce approach can be efficiently used for modular multiplication. In this work we used a two-stage Schoolbook-based multiplier; the multiplication algorithm and the hardware architecture are reported respectively in Algorithm 8 and Figure 3. In this case our crypto-processor is configured to support the NIST P-256 curve only. The 256-bit inputs are split in four parts of 64 bits each and multiplied iteratively. The first stage of the multiplier is composed of two 64×64 bits multipliers and a multiplexer network used to select the proper 64-bit words to be multiplied. Sixteen 64-bit multiplications are required to perform a 256-bit full-word multiplication, so each 64-bit multiplier has to execute eight multiplications. The results are registered into two pipeline registers and processed by the second stage. It is composed of a multiplexer-shifter module that shifts and selects properly the partial products stored into the pipeline registers and one 512-bit adder that sums the content of an accumulation register and the partial products. A finite state machine is used to control the multiplexer networks and to enable the accumulation register. As can be seen in Algorithm 6, modular reduction for NIST P-256 requires six modular additions and four modular subtractions. In our work we implemented it using modular addition/subtraction blocks, computing modular reduction iteratively in three clock-cycles. The latency of the modular multiplier is thirteen clock cycles for a single multiplication but the pipeline reduces the latency to eight cycles on average. Figure 4 shows the data timeline for a modular multiplication for NIST P-256. $M1$ and $M2$ indicate two different modular multiplications. When the pipeline is empty, the first clock cycle is needed to store the operands and both stages of the multiplier are unused; in the second clock cycle only the first stage works, executing two 64-bit multiplications simultaneously. From the third clock cycle to the ninth, both stages are occupied and at the ninth clock cycle the multiplier can store new data and starts a new modular multiplication. In the case where the crypto-processor is configured to support NIST P-521, the architecture of the modular multiplier is similar to the one discussed above. The two 64-bit multipliers are replaced by two 66-bit multipliers and the 512-bit adder by a 1042-bit adder. The 521-bit operands are split in eight parts, requiring thirty-four clock cycles for a 521-bit single multiplication. The reduction algorithm, reported in Algorithm 7, requires only one modular addition executed in one clock cycle. Thirty-five clock cycles are required for a single modular multiplication for the NIST P-521 curve, reduced to thirty-two in pipeline.

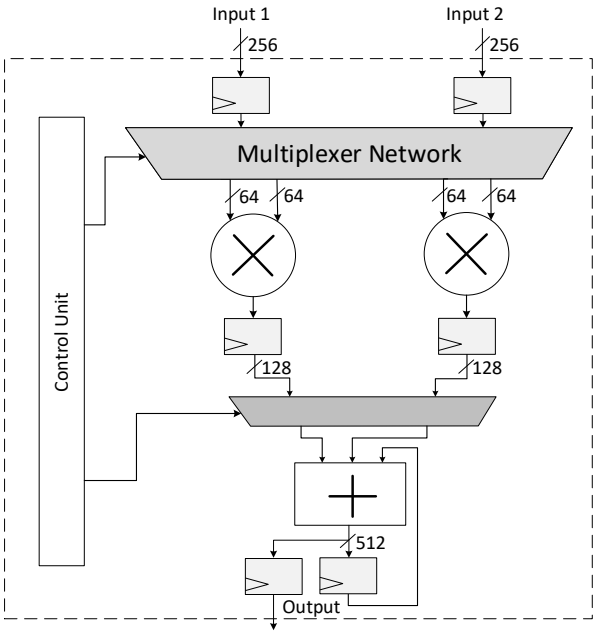

Figure 3. Proposed Hardware Architecture for the 256-bit Multiplier.

Algorithm 8 Schoolbook-Based Multiplication Algorithm.

Input: A, B 256-bit integers such that:
$A = a_3 2^{192} + a_2 2^{128} + a_1 2^{64} + a_0$,
$B = b_3 2^{192} + b_2 2^{128} + b_1 2^{64} + b_0$.
Output: $C = A \times B$

1: $P_0 = a_0 \times b_0$; $P_1 = a_1 \times b_0$; $C = 0$
2: $P_2 = a_0 \times b_1$; $P_3 = a_2 \times b_0$; $C = C + P_0 + P_1 2^{64}$
3: $P_4 = a_1 \times b_1$; $P_5 = a_0 \times b_2$; $C = C + P_2 2^{64} + P_3 2^{128}$
4: $P_6 = a_3 \times b_0$; $P_7 = a_2 \times b_1$; $C = C + P_4 2^{128} + P_5 2^{128}$
5: $P_8 = a_1 \times b_2$; $P_9 = a_0 \times b_3$; $C = C + P_6 2^{192} + P_7 2^{192}$
6: $P_{10} = a_3 \times b_1$; $P_{11} = a_2 \times b_2$; $C = C + P_8 2^{192} + P_9 2^{192}$
7: $P_{12} = a_1 \times b_3$; $P_{13} = a_3 \times b_2$; $C = C + P_{10} 2^{256} + P_{11} 2^{256}$
8: $P_{14} = a_2 \times b_3$; $P_{15} = a_3 \times b_3$; $C = C + P_{12} 2^{256} + P_{13} 2^{320}$
9: $C = C + P_{14} 2^{320} + P_{15} 2^{384}$
10: **return** C

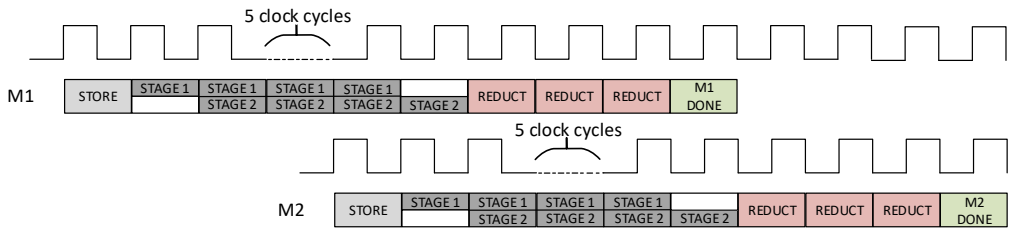

Figure 4. Modular Multiplier Data Timeline for NIST P-256 Curve.

4.3. PA, PD and Modular Inversion

PA and PD operations in Standard projective coordinates have been shown in Equations (1) and (2) of Section 3.3. In this work we implemented two separate hardware modules for PA and PD, as showed in Figure 5. Each of them is composed of one modular multiplier, one modular adder/subtractor module, a multiplexer network and registers bank to store the intermediate results. The scheduling strategy is to parallelize PA and

PD operations and to maximize the parallelism of the field operations that compose them. Considering that the time for computing modular addition/subtraction is negligible with respect to the one to compute modular multiplication, we scheduled the fields operations in order to perform modular addition/subtraction and modular multiplication simultaneously, avoiding stopping the multiplier. Tables 2 and 3 show the scheduling for PA and PD respectively. For PA, 127 and 457 clock cycles (c.c. in Tables 2 and 3) are required in cases of P-256 and P-521 curves, respectively, in contrast to PD, which requires 95 and 329 clock cycles for P-256 and P-521 curves, respectively. Modular inversion has been implemented using Fermat's Little theorem because the presence of two separated modular multipliers allows to easily integrate this technique into the proposed design and to avoid use of a dedicated block, saving area. This theorem allows calculating the modular inverse of an integer a performing a^{p-2}, where p is the modulus. A constant time right-to-left version of square-and-multiply has been used to calculate the modular exponentiation a^{p-2}, reported in Algorithm 9. The modular multiplications given in lines 4 or 6 and 8 of the Algorithm 9 are executed concurrently by the two modular multipliers.

Table 2. PA Operations Scheduling (c.c. stands for clock cycles).

c.c. P-256	c.c. P-521	Modular Multiplication	Modular Add/Sub
0–12	0–34	$X_2 \cdot Z_1$	
8–20	32–66	$X_1 \cdot Z_2$	
16–28	64–98	$Y_2 \cdot Z_1$	$B = X_2Z_1 + X_1Z_2$
24–36	96–130	$B^2 = B \cdot B$	
37–49	131–165	$B^3 = B^2 \cdot B$	
45–57	163–197	$B^2 \cdot X_1Z_2$	
53–65	195–229	$Y_1 \cdot Z_2$	$2 \cdot (B^2X_1Z_2)$
61–73	227–261	$Z_1 \cdot Z_2$	$A = Y_2Z_1 - Y_1Z_2$
69–81	259–293	$A \cdot A$	
82–94	294–328	$A^2 \cdot Z_1Z_2$	
90–102	326–360	$Z_3 = B^3 \cdot Z_1Z_2$	$(A^2Z_1Z_2) - B^3$
98–110	358–392	$B^3Y_1Z_2$	$C = (A^2Z_1Z_2 - B^3) - 2B^2X_1Z_2$
106–118	390–424	$X_3 = B \cdot C$	$(B^2X_1Z_2) - C$
114–126	422–456	$A \cdot (B^2X_1Z_2 - C)$	
127	457		$Y_3 = (A(B^2X_1Z_2 - C)) - (B^3Y_1Z_2)$

Table 3. PD Operations Scheduling (c.c. stands for clock cycles).

c.c. P-256	c.c. P-521	Modular Multiplication	Modular Add/Sub
0–12	0–34	$Y_1 \cdot Z_1$	$X_1 + Z_1$, $X_1 - Z_1$
8–20	32–66	$(X_1 + Z_1) \cdot (X_1 - Z_1)$	$B = Y_1Z_1 + Y_1Z_1$
16–28	64–98	$C = B \cdot Y_1$	$2 \cdot (X_1 + Z_1)(X_1 - Z_1)$
24–36	96–130	$C^2 = C \cdot C$	$A = 3(X_1 + Z_1)(X_1 - Z_1)$
37–49	131–165	$C \cdot X_1$	$2 \cdot C^2$
45–57	163–197	$A^2 = A \cdot A$	$D = 2 \cdot CX_1$
53–65	195–229	$B^2 = B \cdot B$	$2 \cdot D$
61–73	227–261	$Z_3 = B \cdot B^2$	$E = A^2 - 2D$
69–81	259–293	$X_3 = E \cdot B$	$D - E$
82–94	294–328	$A \cdot (D - E)$	
95	329		$Y_3 = (A(D - E)) - (2C^2)$

Algorithm 9 Right-to-left Square-and-Multiply for Modular Exponentation.

Input: $a, x = (x_{n-1}, x_{n-2}, .., x_1, x_0)$
Output: $b = (a^x) mod p$

1: $r_1 = 1, r_2 = a, r_3 = 0$
2: **for** i from 0 to $n-1$ **do**
3: **if** $x[i] = 1$ **then**
4: $r_1 = (r_1 r_2) mod p$
5: **else**
6: $r_3 = (r_1 r_2) mod p$
7: **end if**
8: $r_2 = (r_2)^2 mod p$
9: **end for**
10: **return** r1

4.4. Overall Architecture

PM and DPM are realized based on PA and PD operations. In the case of PM, no precomputed values are used, and the crypto-processor allows selecting which point P has to be used as input between the base point recommended by NIST or another one provided by externally.

In the case of DPM, we used a constant time version of Shamir's Trick reported in Algorithm 3. The scalar k and l together with the point R have to be provided externally while the point P, as in the case of PM, can be selected internally or externally. The overall architecture is reported in Figure 5; a main state machine is used to achieve PM and DPM based on PA and PD in standard projective coordinates. The state machine controls also the operations flow to convert the computed point in the affine domain.

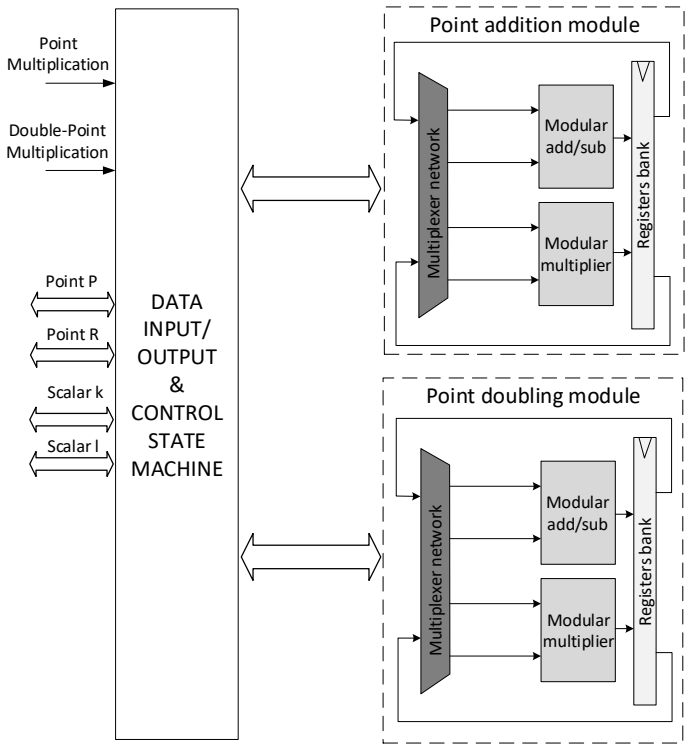

Figure 5. Overall Architecture of the proposed ECC Crypto-Processor.

4.5. FPGA Verification

The crypto-processor has been verified on a Xilinx ZCU106 board. We used the test vectors distributed by NIST for ECDSA in the Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS) [36]. Our crypto-processor has been used to perform the

PM operation required for the generation of a digital signature and the DPM operation required to verify the signature. In addition, our crypto-core has been verified by integration with the NIOS II and other crypto engines (AES and SHA) on Stratix IV FPGA to implement the Hardware Security Module of a WAVE (Wireless Access in the Vehicular Environment) IEEE 802.11p modem for V2X connectivity.

4.6. SPA Assessment through Simulated Approach

To evaluate the proposed SPA countermeasure we used a simulated approach to extract power traces from gate-level netlist without requiring any additional physical circuit or dedicated equipment for power samples acquisition. We implemented three different designs for the algorithms reported in Algorithms 1–3,which are based on the same overall architecture reported in Section 4.4 where the main difference among the three designs is related to the main state and control machine. The steps of our SPA assessment method are reported in Figure 6. The first step requires the logic synthesis of the RTL design, executed using Synopsys Design Compiler [37] with the Standard-cell library Artisan TSMC 7nm (Typical corner case: 0.75V, 85°C). The output of the logic synthesis process is a gate-level netlist which represents an approximation of the physical circuit, and it is used together with the Standard-cell library as input for the gate-level simulations, performed with QuestaSim [38]. The switching activity of the circuit running testbenches is stored in a Value Change Dump (VCD) file during the gate-level simulations and the tool PrimeTime [39] is used to extract the power. Finally, the power trace is parsed and plotted by a Python script. The three different designs have been synthesized at 100 MHz (10 ns of period) and the sampling period has been set to 0.01 ns, in order to generate a fairly dense power trace. Figures 7–9 show the plots of acquired power traces which have been restricted to 7810 ns for reasons of readability and space within the paper.

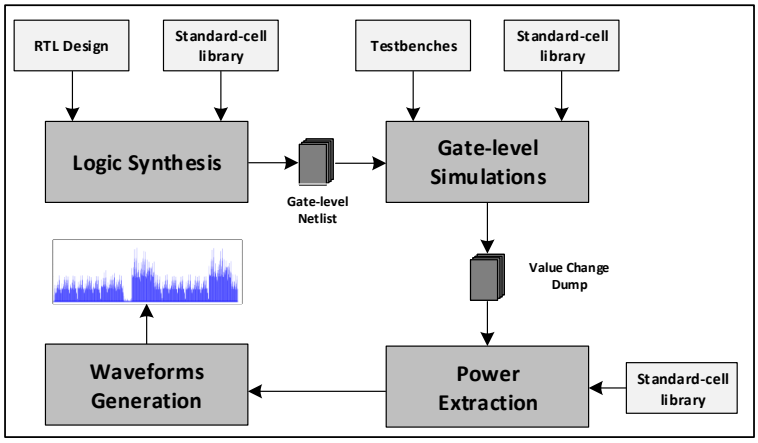

Figure 6. Power Simulation Flow for SPA Assessment.

The traces in Figure 7 are the ones acquired during the execution of Algorithm 1. In this case, the value of the i-th bit of the key k can be easily guessed due to the dissimilarity of the power consumption during the execution of the algorithm. A higher power peak can be easily seen when 1 is encountered (PA and PD operations are executed concurrently), and a lower power peak can be seen when 0 (only PD is executed).

The traces in Figure 8 have been acquired during the execution of Algorithm 2. In this case, the power trace during PA with the point-at-infinity is quite different with respect to PA with real points. Referring to the top left and top right plots in Figure 8, two consecutive PAs with point-at-infinity can be seen. This information can be exploited by an adversary who may be able to understand the fact that the first two least significant digits of the key are different, and may assume the values 01 or 10. Referring to the bottom left and bottom right plots instead, the first and the third PAs are with the point-at-infinity as input. This means that the first two least significant bits of the key are equal, and the third one

is different. In this case, an adversary can hypothesize that the value of the first three least significant bits of the key are 001 or 110. Therefore, in the Double-And-Add-Always algorithm in the Right-to-left version the information leakage of the private key is related to the number of equal bits starting from the least significant part of the key.

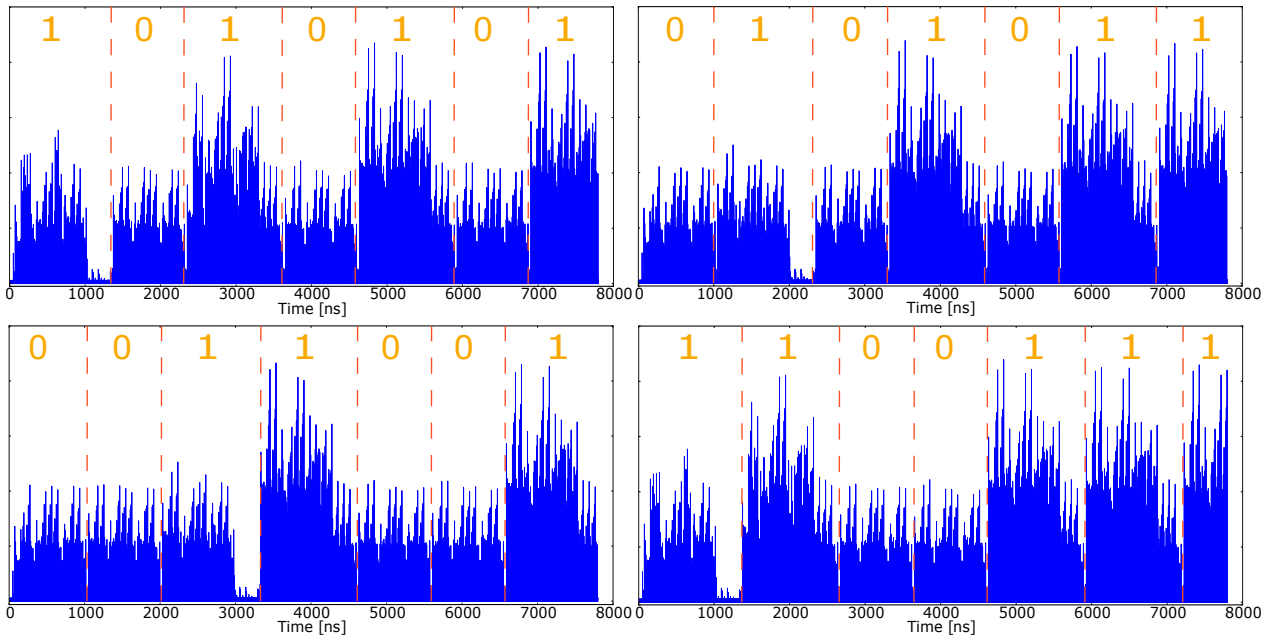

Figure 7. Acquired Power Traces for the Double-and-Add (Algorithm 1) where the least significant part of the key k is (LSB first): -1010101 (**top left**), -0101011 (**top right**), -0011001 (**bottom left**), -1100111 (**bottom right**).

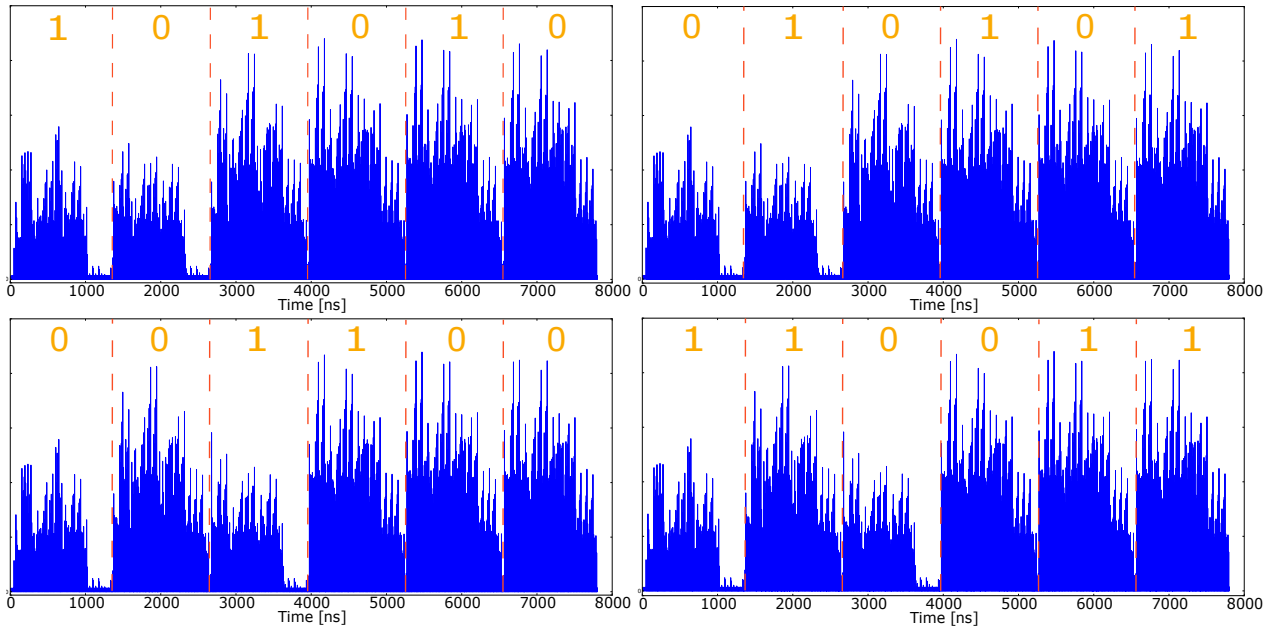

Figure 8. Acquired Power Traces for the Double-and-Add-Always (Algorithm 2) where the least significant part of the key k is (LSB first): -101010 (**top left**), -010101 (**top right**), -001100 (**bottom left**), -110011 (**bottom right**).

The traces in Figure 9 have been acquired during the execution of Algorithm 3. In this case there are no substantial differences among the acquired power traces. It should also be noted that in our simulation environment there are no additional circuits (e.g., processors, communication buses, etc.) that would be present in a real system and would contribute to power consumption, masking any small differences present in the power traces acquired

during the execution of Algorithm 3 and depicted in Figure 9. In any case, the extraction and analysis of real power traces will be carried out to test the effectiveness of the proposed algorithm and the validity of the implemented simulation environment.

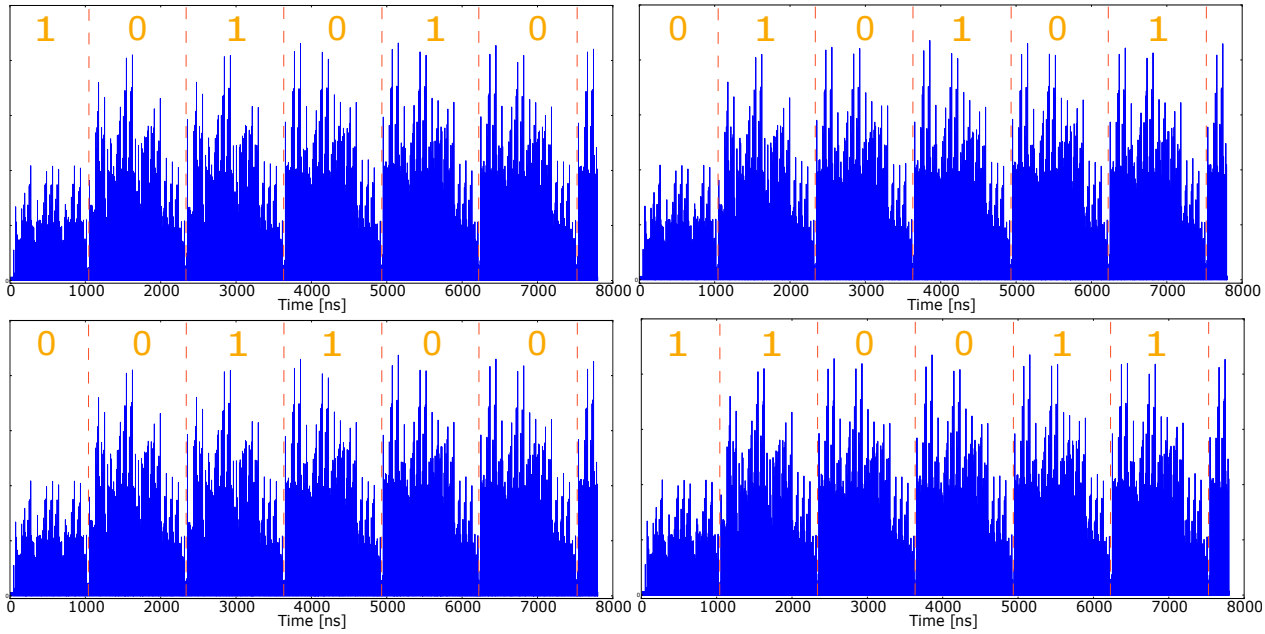

Figure 9. Acquired Power Traces for the Modified Double-and-Add-Always (Algorithm 3) where the least significant part of the key k is (LSB first): -101010 (**top left**), -010101 (**top right**), -001100 (**bottom left**), -110011 (**bottom right**).

5. Results and Comparison

The design described in Section 4 has been synthesized with Design Compiler L-2016.03 both on 45 nm Silvaco and 7 nm Artisan TSMC (Typical corner case: 0.75 V, 85 °C) ASIC standard-cell libraries. Table 4 reports the post-synthesis results; Kcycles column indicates the number of clock cycles required to execute a PM operation; T [µs] column depicts the latency needed for PM at the maximum frequency reported in the column Freq; Column Configuration reports the three possible configurations at the synthesis level for the proposed crypto-processor. In our design, PM requires, respectively, 36.390K and 254.456K clock cycles for NIST P-256 and P-521 elliptic curves in all the configurations. DPM operation requires 61.344K and 430.360K clock cycles for NIST P-256 and P-521, respectively, reducing the computation latency of the 16% respect to executing two separated PM and one PA. On the 45nm Standard-Cell, the maximum frequency is 400 MHz for the P-256 only configuration which decreases to 375 MHz for the other configurations. On the 7 nm Artisan TSMC Standard-Cell, the maximum clock frequency is 1820 MHz for the P-256 only configuration, and 1650 MHz for the other configurations. Table 5 reports the synthesis results on Xilinx ZCU106 board equipped with Zynq UltraScale+ xczu7ev-ffvc1156-2-e MPSoC; columns CLBs and DSPs indicate, respectively, the number of Configurable Logic Block (CLB) and DSP occupied, while T [µs] column depicts the latency for PM at the maximum frequency reported in the column Freq. In 7-series FPGAs, each CLB contains two slices which consist in four 6-input LUTs. No device-dependent optimizations were adopted on the FPGA platform because the goal was only to verify the functionality of the design.

Table 4. Synthesis Results on ASIC Technologies.

Configuration	Process. [nm]	Gate Counts [kGE]	Kcycles	Freq. [MHz]	T [µs]
P-256 only	45	281	36.390	400	90.975
P-521 only	45	407	254.456	375	686.54
P-256/-521	45	447	36.390/257.456	375	97.04/686.54
P-256 only	7	279	36.390	1820	19.99
P-521 only	7	405	257.456	1650	156.03
P-256/-521	7	445	36.39/257.456	1650	22.05/156.03

Table 5. ECC Crypto-Core Resources Utilization on Zynq UltraScale+ xczu7ev-vc1156-2-e MPSoC.

Config.	CLBs	DSPs	Freq. [MHz]	T [µs]
P-256 only	3444	64	150	242
P-521 only	5689	64	120	2145
P-256/-521	6575	64	110	330/2340

5.1. Discussion and Comparison

Several ECC processors proposed in the literature are implemented in different FPGA platforms or ASIC technologies, making the process of comparison and benchmarking extremely complicated. For this reason, in order to make a fair comparison with previous works, in this paper we present a comparison among our synthesis results on 45 nm and other ECC systems synthesized on ASIC technologies from 55 nm to 130 nm. The results are reported in Table 6. The column denoted with T [µs] indicates the time needed to execute a PM, and the column AT indicates the area-time product that is normalized into 45 nm to compare all the reported designs implemented in different processes.

Table 6. Comparison among previous works for 256-bit PM.

Ref.	Process. [nm]	Gate Counts [kGE]	Primes	Kcycles	Freq. [MHz]	T [µs]	AT	SPA Assessment
Our	45	281	P-256	36.390	400	90.97	1	Simulated approach
[31]	55	187	Dual-Field	–	316	1450	8.68	Power extraction
[24]	65	3500	P-256	2.35	188	12.5	1.18	–
[20]	65	447	P-256	397.3	546.5	730	8.84	–
[32]	90	540	256-bit	22.3	185	120	1.27	Theoretical
[33]	130	122	256-bit	340	556	1010	1.67	–
[23]	130	77.1	256-bit	–	200	860	0.9	–
[22]	130	57.05	256-bit	610	150	4070	3.14	Theoretical

The work in [31] is a Dual-Field ECC processor that supports an arbitrary elliptic curve. The result of this work is more flexible with respect to our design but achieves lower performance in terms of speed and AT. The design in [24] is faster in respect to ours, but it has higher AT and no SPA resistance has been guaranteed. The processor in [20] supports both NIST P-224 and P-256 elliptic curves. Their design requires more area with respect to our implementation and achieves a higher AT without any protection against power attacks. The work in [32] performs a PM in 120 µs with 540K gate counts. Their design requires less c.c. with respect to our work, but the area consumption is higher together with the AT product. In this case, no experimental results have been provided to test the SPA resistance, but the authors claim their solution works well. Works in [23,33] do not implement any countermeasure against side-channel attacks. The design in [33] reaches good performance in terms of area occupation but lower speed with respect to our work. The modular square method proposed in [23] allows good performance and an AT similar to the one achieved by our crypto-processor. The work in [22] is a low hardware

consumption design for ECC. The area consumption in fact is lower with respect to our work, but both speed and AT are higher. The adopted SPA countermeasure has not been tested and is only theoretical. In addition, our crypto-processor can be configured at a synthesis level to also support the NIST P-521 elliptic curve, providing a security level of 256 bits. Using the proposed crypto-processor on 7 nm Artisan technology for running ECDSA algorithms, up to 50k and 29k digital signatures per second can be generated and verified on the NIST P-256 curve. These results are up to four orders of magnitude better with respect to the ones achieved on Cortex-M processors reported in [18], where the power consumption comprises between 118.5 mW and 281.8 mW. In our work we estimated the power consumption of our crypto-processor (configured to support only NIST P-256 curve), which is around 49 mW @ 400 MHz and 102 mW @ 1.82 GHz, respectively, in 45 nm Silvaco at 1.1 V and 7 nm Artisan at 0.75 V. These results have been extracted by means of the PrimeTime tool.

6. Conclusions

In this paper, we proposed a fast and configurable ECC crypto-processor for NIST P-256/-521 elliptic curves. It has been synthesized both on 45nm Silvaco and 7nm Artisan TSMC technologies, and verified on a Xilinx ZCU106 board with official NIST test vectors for ECDSA. The presented processor can be used to accelerate ECDH, ECMQV, ECIES and ECDSA algorithms based on ECC. A simulation environment to extract and evaluate the power traces during the execution of PM has been implemented, allowing the design of a modified version of a Double-And-Add-Always algorithm as a countermeasure against SPA and timing attacks. This work is part of the early development phase for the architecture of the ECC crypto-accelerator that, together with other crypto-engines that we are also designing (i.e., AES, SHA, RNG), will be integrated into the HSM of the European Processor Initiative chip. Synthesis results on a 45nm Standard-Cell show that performance in terms of speed and area consumption are aligned with the state-of-the-art with an optimal AT. On 7nm technology the speed performance in absolute value outperforms most of the previous works. Our work is the first contribution in literature with synthesis results on 7 nm technology. Although NIST and other standardization institutes are running a standardization process for post-quantum public-key algorithms, currently ECC is one of the public-key system most often adopted for key agreement, encryption/decryption and digital signatures services.

Author Contributions: Conceptualization and methodology, S.D.M., L.B., L.C. and P.N.; validation, S.D.M. and L.C.; project administration and founding acquisition, S.S. and L.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially funded by the European Processor Initiative (EPI) project, under grant agreement No. 826646.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
2. Stallings, W. Symmetric Ciphers. *Cryptography and Network Security: Principles and Practice*, 5th ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2010.
3. Lenstra, A.K.; Verheul, E.R. Selecting cryptographic key sizes. *J. Cryptol.* **2001**, *14*, 255–293. [CrossRef]
4. Miller, V.S. Use of elliptic curves in cryptography. In Proceedings of the Conference on the Theory and Application of Cryptographic Techniques, Linz, Austria, 9–11 April 1985; Springer: Berlin/Heidelberg, Germany, 1985; pp. 417–426.
5. Koblitz, N. Elliptic curve cryptosystems. *Math. Comput.* **1987**, *48*, 203–209. [CrossRef]
6. Group, I.W. *IEEE 1363-2000: Standard Specifications for Public Key Cryptography*; IEEE Standard: New York, NY, USA, 2000; Volume 10017.
7. NIST. *FIPS 186-4–Digital Signature Standard (DSS)*; Information Technology Laboratory, National Institute of Standards and Technology: Gaithersburg, MD, USA, 2013.

8. ANSI. X9.62:Public key cryptography for the financial services industry. In *The Elliptic Curve Digital Signature Algorithm (ECDSA)*; American National Standards Institute: Washington, DC, USA, 2005.
9. Standards for Efficient Cryptography SEC 1, SEC 2: Elliptic Curve Cryptography; Certicom Research: Mississauga, ON, Canada, 2009. Available online: https://www.secg.org (accessed on 10 May 2021).
10. Fips, P. *186-2. Digital Signature Standard (dss)*; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2000; Volume 20, p. 13.
11. Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.3, RFC, Volume 8446. 2018. Available online: https://www.rfc-editor.org/info/rfc8446 (accessed on 10 May 2021).
12. Burgin, K.; Peck, M. Suite B Profile for Internet Protocol Security (IPsec); Technical Report, RFC 6380. 2011. Available online: https://www.rfc-editor.org/rfc/rfc6380.html (accessed on 7 June 2021).
13. Group, I.W. *IEEE Standard for Wireless Access in Vehicular Environments-Security Services for Applications and Management Messages*; IEEE Std 1609.2-2016; IEEE: New York, NY, USA, 2016; pp. 1–240.
14. ETSI Technical Specification. ETSI TS 103 097 v1. Intelligent Transport Systems (ITS); Security; Security Header and Certificate Formats, Standard, TC ITS, 2013. Available online: https://www.etsi.org/deliver/etsi_ts/103000_103099/103097/01.04.01_60/ts_103097v010401p.pdf (accessed on 17 April 2021).
15. Naveed Aman, M.; Taneja, S.; Sikdar, B.; Chua, K.C.; Alioto, M. Token-Based Security for the Internet of Things With Dynamic Energy-Quality Tradeoff. *IEEE Internet Things J.* **2019**, *6*, 2843–2859. [CrossRef]
16. AlMajed, H.; AlMogren, A. A Secure and Efficient ECC-Based Scheme for Edge Computing and Internet of Things. *Sensors* **2020**, *20*, 6158. [CrossRef] [PubMed]
17. Baldanzi, L.; Crocetti, L.; Di Matteo, S.; Fanucci, L.; Saponara, S.; Patrice, H. Crypto accelerators for power-efficient and realtime on-chip implementation of secure algorithms.In Proceedings of the 2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Genoa, Italy, 27–29 November 2019.
18. Ledwaba, L.P.; Hancke, G.P.; Venter, H.S.; Isaac, S.J. Performance costs of software cryptography in securing new-generation Internet of energy endpoint devices. *IEEE Access* **2018**, *6*, 9303–9323. [CrossRef]
19. Verri Lucca, A.; Mariano Sborz, G.A.; Leithardt, V.R.Q.; Beko, M.; Albenes Zeferino, C.; Parreira, W.D. A Review of Techniques for Implementing Elliptic Curve Point Multiplication on Hardware. *J. Sens. Actuator Netw.* **2021**, *10*, 3. [CrossRef]
20. Hossain, M.S.; Kong, Y.; Saeedi, E.; Vayalil, N.C. High-performance elliptic curve cryptography processor over NIST prime fields. *IET Comput. Digit. Tech.* **2016**, *11*, 33–42. [CrossRef]
21. Awaludin, A.M.; Larasati, H.T.; Kim, H. High-Speed and Unified ECC Processor for Generic Weierstrass Curves over GF(p) on FPGA. *Sensors* **2021**, *21*, 1451. [CrossRef] [PubMed]
22. Hu, X.; Zheng, X.; Zhang, S.; Cai, S.; Xiong, X. A Low Hardware Consumption Elliptic Curve Cryptographic Architecture over GF(p) in Embedded Application. *Electronics* **2018**, *7*, 104. [CrossRef]
23. Li, B.; Lei, B.; Zhang, Y.; Lei, S. A Novel and High-Performance Modular Square Scheme for Elliptic Curve Cryptography Over GF (*p*). *IEEE Trans. Circuits Syst. II: Express Briefs* **2018**, *66*, 647–651. [CrossRef]
24. Liu, J.; Cheng, D.; Guan, Z.; Wang, Z. A High Speed VLSI Implementation of 256-bit Scalar Point Multiplier for ECC over GF (p). In Proceedings of the 2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR), Shenyang, China, 24–27 August 2018; pp. 184–191.
25. Hankerson, D.; Menezes, A.J.; Vanstone, S. *Guide to Elliptic Curve Cryptography*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2006.
26. European Processor Initiative (EPI). Available online: https://www.european-processor-initiative.eu (accessed on 10 May 2021).
27. Nannipieri, P.; Di Matteo, S.; Baldanzi, L.; Crocetti, L.; Belli, J.; Fanucci, L.; Saponara, S. True Random Number Generator Based on Fibonacci-Galois Ring Oscillators for FPGA. *Appl. Sci.* **2021**, *11*, 3330. [CrossRef]
28. Baldanzi, L.; Crocetti, L.; Falaschi, F.; Bertolucci, M.; Belli, J.; Fanucci, L.; Saponara, S. Cryptographically Secure Pseudo-Random Number Generator IP-Core Based on SHA2 Algorithm. *Sensors* **2020**, *20*, 1869. [CrossRef] [PubMed]
29. Nannipieri, P.; Bertolucci, M.; Baldanzi, L.; Crocetti, L.; Di Matteo, S.; Falaschi, F.; Fanucci, L.; Saponara, S. SHA2 and SHA-3 Accelerator Design in a 7 nm Technology Within the European Processor Initiative. *Microprocess. Microsyst.* **2021**. [CrossRef]
30. Silvaco PDK 45 nm Open Cell Library. Available online: https://si2.org/open-cell-library (accessed on 30 July 2021).
31. Liu, Z.; Liu, D.; Zou, X. An efficient and flexible hardware implementation of the dual-field elliptic curve cryptographic processor. *IEEE Trans. Ind. Elec.* **2016**, *64*, 2353–2362. [CrossRef]
32. Chung, S.C.; Lee, J.W.; Chang, H.C.; Lee, C.Y. A high-performance elliptic curve cryptographic processor over GF (p) with SPA resistance. In Proceedings of the 2012 IEEE International Symposium on Circuits and Systems, Seoul, Korea (South), 20–23 May 2012; pp. 1456–1459.
33. Chen, G.; Bai, G.; Chen, H. A High-Performance Elliptic Curve Cryptographic Processor for General Curves Over GF (p) Based on a Systolic Arithmetic Unit. *IEEE Trans. Circuits Syst. II: Express Briefs* **2007**, *54*, 412–416. [CrossRef]
34. Brier, E.; Joye, M. Weierstraß elliptic curves and side-channel attacks. In Proceedings of the International Workshop on Public Key Cryptography, Paris, France, 12–14 February 2002; pp. 335–345.
35. Pontie, S.; Maistri, P.; Leveugle, R. Dummy operations in scalar multiplication over elliptic curves: a tradeoff between security and performance. *Microprocess. Microsyst.* **2016**, *47*, 23–36. [CrossRef]

36. Hall, T.A. Keller, S.S. *Elliptic Curve Digital Signature Algorithm (ECDSA) Validation System (ECDSA2VS)*; NIST Information Technology Laboratory: Gaithersburg, MD, USA, 2014.
37. Synopsys Design Compiler. Available online: https://www.synopsys.com/support/training/rtl-synthesis/design-compiler-rtl-synthesis.html (accessed on 22 June 2021).
38. QuestaSim. Available online: https://eda.sw.siemens.com/en-US/ic/questa/simulation/advanced-simulator/ (accessed on 22 June 2021).
39. PrimeTime. Available online: https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html (accessed on 22 June 2021).